

Engineering the Technical Interview

Whitepaper March 2020

CHALLENGE

The average interview is inconsistent and under-achieves on hiring targets

Many software engineers who interview candidates feel that they know how to conduct an interview that will predict the candidate's performance in the onsite interview loop and on the job. Yet, most teams fail to conduct enough interviews with the quality and consistency required to hire the software engineers they need, creating an Interview Gap.



Only 38% of engineering leaders are confident in hitting their 2020 hiring targets



67% of engineering leaders say 'very few people at my company know how to conduct technical interviews'

This paper explores how companies can equip developers and hiring teams with a framework for conducting predictive technical interviews as part of a consistent, structured process.

SOLUTIONS

Structuring the interviewing and hiring process

Companies are able to reach hiring targets when they conduct more interviews with structure and consistency while tracking results and using feedback to optimize the process. They can achieve this by professionalizing the interview and hiring process through the implementation of six best practices that make the process fair and highly predictable: defining hiring competencies, improving interview formats and questions, conducting interviews with standards for interviewer performance, measuring candidate performance consistently, establishing a clear hiring bar and analyzing hiring data to evolve research.

RESULTS

Improved interview experience that reaches more of the hiring target

Organizations who streamline and structure the interviewing and hiring process are able to gain visibility into a previously opaque process, increase hiring yield, and improve candidate experience.

Engineering the technical interview

The promise of any interview is clear: to connect talent to opportunity. While companies need the interview to build teams, candidates rely on it to take the next step in their career. Hiring managers can develop an interviewing and hiring process that generates reliable outcomes by using this framework and best practices:

1. Define and accurately assess hiring competencies
2. Improve formats and questions
3. Conduct the interviews professionally
4. Measure candidate performance consistently
5. Establish a clear hiring bar
6. Analyze hiring data to evolve approach

Define and accurately assess hiring competencies

For each open software engineering role, your team should assign another developer to review the job description and responsibilities, then align these elements to specific competencies.

To get a more predictive signal, interviewers and hiring teams should ask themselves:

- Are the competencies relevant to success on the job?
- Am I measuring each competency separately?

Establishing relevant competencies

The output of this process hinges on competencies that are predictive of job success. One helpful way to determine what competencies to assign to a role is to think about how the person is going to be evaluated in their review at the end of the year. Will the performance review focus on broad aptitudes such as language skills, reading comprehension, US cultural awareness or easily acquired knowledge such as search ability? For a senior software engineering role, the competencies are more likely to include specific language skills such as Java, demonstrated ability to understand and articulate business logic complexity, and completing an architecture review.

It's also imperative that candidates understand what competencies you're assessing. What are the mechanisms you have in place to clearly communicate to the candidate that they are being assessed for code quality, optimality vs. speed, or the ability to deal with ambiguity? Hiring managers should clearly define the competencies they're looking for, and interviewers should clearly articulate those competencies to candidates so they know whether to deliver complete code or an outline, if a brute-force solution is good enough, or if they need to take the next step to optimize it.

Measure each competency separately

Interviewers often use a single question to measure several competencies at once: Can this person code quickly? Can they think out loud? Can they speed-read four paragraphs of English text? Does the candidate know the rules to the same board games? This is detrimental because it doesn't isolate a variable to assess the competency and because it introduces noise through measuring competencies that may not prove relevant to the job.

For example, candidates are often required to test their code in an interview. Asking them how they would test it, as opposed to embedding it into a coding exercise, will isolate code testing as a skill. This approach to isolation followed by evaluation boosts the value of each interview segment and makes it possible to optimize the segments separately. It also enables interviewers to exert greater control over measuring the desired competency.

Improve formats and questions

Non-structured interview questions may exclude qualified candidates

One example of a non-structured interview question requires candidates to illustrate a diagram of a well-known software system on the board. This question is so widely used it has become an industry cliché. Unfortunately, the question fails to assess the skills of candidates who have not been previously exposed to the definitions of the boxes and lines on the board. This can introduce unintended bias for candidates from non-traditional backgrounds. Additionally, it tests for a specific system design competency and it may not be relevant to the role.

The structure of the interview question matters as much as its objective of accurately evaluating the candidate's knowledge. A well-structured question will make clear to the candidate what is expected of them and reduce the risk of false negatives.

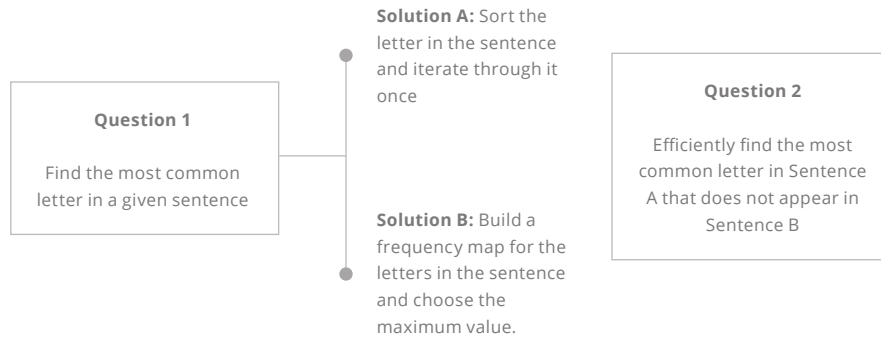
Easily understood questions yield the most signal

Ambiguous questions are quite popular. They expect candidates to have a shared understanding of a phrase that could have multiple interpretations but is also particular to a certain in-group, like an Ivy League college. However, they don't always reveal relevant skills. Instead, teams should rely on easily understood questions to yield the most signal.

For example, an effective system-design question is based on a simple, concrete mechanism that requires the candidate to scrutinize specific aspects of the system's behavior. This way, the expectations are clear and the number of possible answers is limited.

Unbalanced interviews can give candidates an unfair (dis)advantage

As illustrated in the diagram below, a seemingly ordinary multi-part interview question can actually be unbalanced:



A candidate is asked to find the most common letter in a sentence as part of an introductory question. In the follow-up, they must find the most common letter in sentence A that doesn't appear in sentence B. Two possible solutions to Question 1 are displayed. The two solutions have roughly equal merit and, initially, there's no compelling reason to choose one or the other. But once faced with Question 2, the candidate who chose Solution B has a head start over the one who chose Solution A.

While the example may seem contrived, practice has shown that most multi-part questions produce unwanted downstream effects. This is why interviews should use questions that are conceptually linked but don't share implementation code.

Conduct the interviews professionally

Most teams haven't made interviewing of software engineers a job dedicated to specific team members. However, you may consider making it one in order to ensure that the success of the interview relies on the same rigor as other engineering disciplines. Teams aiming to do this can follow these steps to establish this role and conduct interviews professionally.

- **Adhere to interview guidelines:** Interview length, content, communication, scoring rubrics, and approaches to completing write ups are all structured and standardized
- **Create defined, relevant, and transparent competencies:** As mentioned earlier in this paper, competencies should be defined and aligned to each role, while questions assess one competency at a time, and ambiguity is avoided—this serves to increase signal in each interview
- **Establish mechanisms for accountability:** Just like code review, interviews should be quality checked—this can also help reduce bias

Lastly, a professionally conducted technical interview ought to be able to put candidates at ease. Teams that select interviewers with a baseline of soft skills and interest in interviewing can add a level of empathy and kindness that reduces anxiety for candidates and increase signal. Consider exposing interviewers to the candidate experience—especially if they are more tenured.

Measure candidate performance consistently

Companies often use a wide spectrum of subjective attributes and grades to evaluate candidates, but without a structured scoring rubric, feedback becomes impossible to aggregate and compare from interviewer to interviewer. Structured scoring rubrics aligned to competencies allow for more consistent reporting around areas like solution optimality, completion, handholding, and debugging.

Establish a clear hiring bar

Establishing a clear hiring bar requires evaluating outcomes of the scoring rubric and mapping them to the results of each onsite loop. When similar competencies are identified and candidate success similarly demonstrated, then the hiring bar is well represented in the technical interview and the onsite.

To identify a clear hiring bar, interviewers should work together to determine what level of performance is necessary in each competency for job success; and how to arrive at this decision from the output of each interview. Many companies struggle with evaluating interview performance because, much like the interview, round tables, debriefs and hiring committees are often opaque. As a result, documenting the conclusions reached and the standards by which they are reached can be difficult and time consuming. For this reason, it's crucial to introduce a meaningful inquiry about these processes on a routine basis.






Understanding your hiring bar relative to the market will help you gauge whether or not it is too high, which can lead to rejecting candidates who can be successful in the role. This can be done by monitoring and evaluating the career path of the rejected candidates. Are peer companies snapping them up? Or worse, companies considered to have a higher bar? Answers to these questions will reveal if the hiring bar is needlessly filtering candidates out. An accurately calibrated hiring bar will filter more candidates in.

Analyze hiring data to evolve approach

At this final stage, interviewers' only task is to apply structured data so they can optimize the process based on the evaluation of outcomes. An interview process with enough data structure to draw meaningful conclusions enables hiring managers to make optimizations unavailable to their competitors.

The value of assigning someone to ensure that the data is consistent and regularly tracked reduces pathological data quality in the applicant tracking system (ATS). Questions such as, "Did the candidate reject you or did you reject them?" can be telling of the defects in the interview process. It's not unusual to find a data quality issue in 40% or more of the records in an organization's ATS.

Key takeaways

-  **Give interviewing an owner.**
By assigning an engineer to optimize each part of the interview you create accountability for the consistency of that section.
-  **Be intentional and clear with questions.**
Clearly identify the competencies that matter to the role, communicate those competencies to the candidate, and measure them one at a time.
-  **Centralize interviewing and training.**
Inconsistent interviewers can negatively impact your hiring yield and candidate experience. Don't just train on the general guidelines of interviewing—educate interviewers about the specific questions they should ask and establish consistent standards.
-  **Find a way to audit your interviews.**
Even if you can't record your interviews, use shadows, and give feedback to non-predictive interviewers or those who don't follow a structure. Scrutinize your questions rigorously and discard the ones with too many flaws.
-  **Build your process for data collection from day one.**
Interviewing processes with intention and structure from can be optimized for a variety of desirable characteristics, like speed, candidate experience, and predictiveness.

Ready to discover more?

To talk to Karat about how we deliver technical interviews that are consistently high in quality, visit us at [karat.com](https://www.karat.com)